The valve location problem in simple network topologies

The Valve Location Problem in Simple Network Topologies

Hans L. Bodlaender

Alexander Grigoriev

Nadejda V. Grigorieva

Albert Hendriks

Department of Information and Computing Sciences,
Utrecht University

Technical Report UU-CS-2007-019

www.cs.uu.nl
ISSN: 0924-3275

The Valve Location Problem in Simple Network Topologies

Hans L. Bodlaender* Alexander Grigoriev† Nadejda V. Grigorieva‡
Albert Hendriks§

Abstract

To control possible spills in liquid or gas transporting pipe systems, the systems are usually equipped with shutoff valves. In case of an accidental leak these valves separate the system into a number of pieces limiting the spill effect. In this paper, we consider the problem, for a given edge-weighted network representing a pipe system and for a given number of valves, to place the valves in the network in such a way that the maximum possible spill, i.e. the maximum total weight of a piece, is minimized. We show that the problem is NP-hard even if restricted to any of the following settings: (i) for series-parallel graphs and hence for graphs of treewidth two; (ii) if all edge weights equal one. If the network is a simple path, a cycle, or a tree, the problem can be solved in polynomial time. We also give a pseudo-polynomial time algorithm and a fully polynomial approximation scheme for networks of bounded treewidth.

Keywords: Valve location problem; computational complexity; bounded treewidth; dynamic programming; binary search.

1 Introduction

In this paper, we consider a combinatorial problem that arose from a number of applications connected to operations and maintenance of liquid- or gas-transporting pipe systems; for applications related to the long oil and gas pipelines see e.g. [9]; for applications in water supply engineering see [14]. A pipeline is the most efficient and environmentally friendly way to transport hazardous liquids and gases, e.g. crude oil or natural gas, over land. In normal daily operations, pipelines do not produce any pollution. However, due to

*Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, NL–3508 TB Utrecht, The Netherlands, hansb@cs.uu.nl

†Maastricht University, Quantitative Economics, P.O.Box 616, NL–6200 MD Maastricht, The Netherlands

‡Institute of Power Resources Transport (IPTER), 144/3, pr. Octyabrya, Ufa-450055, Russia

§Department of Information and Computing Sciences, Utrecht University. alberthendriks@gmail.com

external factors or pipe corrosion, accidents on pipelines sometimes happen and the accidental damage can be substantial. To control possible spills, every pipe system is usually equipped with special shutoff valves. Whenever the pipe system is depressurized, the valves automatically and instantly separate the pipe system into *pieces*. Therefore, the quantity of hazardous liquid or gas potentially leaving the system equals the total length of the pipes in the damaged piece of the system separated by shutoff valves. In the application at hand, there is a given edge-weighted network representing a pipe system and a given number of valves that can be placed in the vertices of the network. We want to solve the following problem: find a valve location in the network that minimizes the maximum total weight of a piece separated by shutoff valves.

This paper is organized as follows. In Section 2, we give a precise graph theoretic formulation of the problem. In Section 3, we show that using dynamic programming the problem can be solved in polynomial time on simple network topologies: paths, cycles and trees. In Section 4, we consider a more general case, namely the graphs of bounded treewidth. For these graphs, we give a pseudo-polynomial time dynamic programming algorithm, and then we turn this algorithm into a fully polynomial approximation scheme (FPTAS). Finally, in Section 5, we discuss the complexity of the problem. Here, we show that the problem is NP-hard even for series-parallel graphs and hence for graphs of treewidth at most two. We also show that the unweighted version of the problem, i.e. the problem where all edge weights equal one, is also NP-hard.

2 Graph Theoretic Formulation

The problem can be formulated in graph theoretic terms in a natural way. Let $G = (V, E)$ be an undirected graph representing a pipe network. Edges of the graph represent pipes. Let $\omega_e \in \mathbb{Z}^+$ denote the length of pipe $e \in E$. Vertices of the graph represent connection points between the pipes. Let k be a number of valves to be installed. We assume that a valve can be located in any vertex $v \in V$.

Consider a set of vertices $V' \subseteq V$. If we use V' as valve locations, we use $|V'|$ valves, and partition G into *pieces* as follows. The set of edges E is partitioned into sets with two edges in the same set of the partition if and only if they are on a path in G that does not contain a valve. Thus, E is partitioned into subsets $E_1, E_2, \ldots, E_S$ where edges in E_s, $1 \leq s \leq S$, form a connected component in G called a piece, and for any two subsets, E_s and E_t, the set of endpoints in E_s intersects the set of endpoints in E_t only in elements of V'. The *cost* of V', denoted $W_{\max}(V')$, is

$$W_{\max}(V') = \max_{1 \leq s \leq S} \sum_{e \in E_s} \omega_e,$$

i.e., the maximum total length of a piece or the maximum *spill*.

The VALVE LOCATION problem then is to find a set of valve locations V' in G with minimum cost $W_{\max}(V')$. In other words, we have to find a k-elementary separator in G

such that the maximum length connected component is minimized. We consider also the unweighted version of the problem where $\omega_e = 1$ for all $e \in E$.

Throughout the paper, n denotes the number of vertices in G, $\omega_{\max}$ the maximum length of an edge:

$$\omega_{\max} = \max_{e \in E} \omega_e,$$

and ω_Σ the total length of all edges:

$$\omega_\Sigma = \sum_{e \in E} \omega_e.$$

Clearly, the maximum spill is yet another network vulnerability measure. This concept is very close to many other known vulnerability measures, e.g. *vertex integrity* of a graph defined as $I(G) = \min\{|S| + m(G-S) : S \subset V\}$, where $m(H)$ denotes the maximum order of a component of H, see [2, 3]; *minimum balanced separator* defined as a minimum order separator S such that the maximum component in $G-S$ contains at most βn vertices for a given $0 < \beta < 1$, see [1, 7, 13]; and some other, see e.g. [12]. The key difference between the maximum spill and the known vulnerability measures is that the maximum spill measures vulnerability of a graph in terms of the total edge weight (or length) of a component when all other measures are related to the maximum order (number of vertices) in a component. Of course, practical suitability of a certain measure depends heavily on applications.

Throughout the paper, we measure run time of the algorithms using the widely accepted convention that we can do an addition or multiplication of two integers in $O(1)$ time. If we want to count bit operations, we must multiply run times by a factor $\log \omega_\Sigma$.

3 Simple Networks: Paths, Cycles and Trees

In this section, we give dynamic programming algorithms to solve the problem in simple network topologies: paths, cycles and trees.

3.1 The Valve Location Problem on a Path

We first consider the VALVE LOCATION problem on a path. This simple case appears frequently in the practical settings of the long oil pipelines, and thus is of practical relevance; see [9].

We have two different exact algorithms. One uses 'text book' dynamic programming, the other one uses a binary search for the optimum, and a 'text book' greedy decision algorithm.

Proposition 1 *The* VALVE LOCATION *problem on a path can be solved in* $O(kn^2)$ *time.*

Proof. Without loss of generality, we assume that all vertices in the path are successively numbered by $1, 2, \ldots, n$, and the successive edges on the path have lengths $\omega_1, \omega_2, \ldots, \omega_{n-1}$.

Let $f(v, j)$ denote the min-max spill for the first v edges, if we can use j valves among the first v vertices, i.e. the minimum over all possible positions of j valves on the path formed by the first v vertices of the maximum length of a subpath (a piece) with no internal vertices accommodating a valve. One can directly observe that the following recursive formulation holds for all v, $1 \leq v \leq n$, and all j, $1 \leq j \leq k$:

$$f(v, j) = \min_{1 \leq u \leq v} \max\{f(u, j-1), \sum_{u \leq w \leq v-1} \omega_w\}. \tag{1}$$

In addition, we have for all v, $1 \leq v \leq n$:

$$f(v, 0) = \sum_{1 \leq w \leq v-1} \omega_w. \tag{2}$$

As a preprocessing step, we tabulate in $O(n^2)$ time all values $\sum_{u \leq w \leq v} \omega_w$, for all pairs u and v, $1 \leq u \leq v \leq n-1$. Then, using Equation (2) in the first step and recursively applying Equation (1) in step j, $1 \leq j \leq K$, we obtain at step K the optimal min-max spill $f(n-1, K)$. Then, in the same amount of time, we can also construct a placement of the valves that gives the optimal spill. □

A different algorithm is obtained by using a binary search for the optimal value, checking each value with a greedy algorithm.

Proposition 2 *Given a path and a value L, we can decide in $O(n)$ time if there is a solution to the* VALVE LOCATION *problem with k valves with cost at most L.*

Proof. The following simple greedy algorithm suffices. Move along the path from left to right, and put a valve whenever adding one additional edge to the current piece would create a piece of total length more than L. If the last piece has length at most L after we placed all k valves, or if we reach the end of the path before using all k valves, we succeed. Otherwise, there is no solution with cost at most L. □

The same argument also gives the minimum number of valves needed to guarantee a cost that is at most L. Using binary search for the optimal value in the range of integers between 0 and ω_Σ, we directly obtain the following result.

Corollary 3 *For a given path, we can solve the* VALVE LOCATION *problem in $O(n \log \omega_\Sigma)$ time.*

It is also possible to construct a fast 2-approximation algorithm using a greedy strategy for paths. As with k valves, we divide the path in at most $k+1$ pieces, and there always will be a piece containing the maximum length edge, $\max\{\omega_\Sigma/(k+1), \omega_{\max}\}$ is a lower bound for the optimal value.

Consider the following greedy algorithm. Write $A_p = \omega_\Sigma/(k+1)$. Move through the path from left to right, and put a valve as soon as we have a piece of size at least A_p since the last valve (or the start of the path, in case we place the first valve). As each piece, except the last one, has size at least A_p, the length of the last piece is at most $\omega_\Sigma - kA_p \leq A_p$. For each other piece, its total length is less than A_p plus the size of the longest edge, hence is at most $A_p + \omega_{\max}$, and hence at most twice the lower bound $\max\{A_p, \omega_{\max}\}$. Thus, this greedy algorithm is a 2-approximation. It clearly runs in $O(n)$ time. We summarize the obtained results in the following proposition.

Proposition 4 *The* VALVE LOCATION *problem on a path admits a 2-approximation algorithm that uses $O(n)$ time.*

We can sharpen Proposition 4 when $\omega_{\max} \geq 3A_p$.

Proposition 5 *Consider the* VALVE LOCATION *problem on a path. Let k be the number of valves. If $\omega_{\max} \geq 3\omega_\Sigma/(k+1)$, then the optimal solution has cost $\omega_{\max}$. An optimal solution can be found in this case in $O(n)$ time.*

Proof. Clearly, $\omega_{\max}$ is a lower bound for the optimal value. Again, write $A_p = \omega_\Sigma/(k+1)$.

Consider the following algorithm. We visit the vertices on the path from left to right, and put a valve on a vertex, when the piece since the last placed valve (or, for the first valve, since the start of the path) has total length at least A_p, or when the next edge has a length that is at least $2A_p$. We end if we placed k valves, or have arrived at the end of the path.

Note that this effectively means that we put at both endpoints of an edge with length at least $2A_p$ a valve (except at the endpoints of the path itself.)

Each piece now has a total length that is at most $\omega_{\max}$. We can see this with case analysis.

- Suppose we placed k valves. The last piece has total length at most A_p. Consider the first k pieces. Each of these has total length at least A_p, or has a total length less than A_p, but then the next piece has length at least $2A_p$. So, the total length of the first k pieces is at least kA_p, and hence the last piece has total length at most $\omega_\Sigma - kA_p = A_p$.
- A piece that is not the last piece that contains at most one edge has length at most $\omega_{\max}$.
- Consider a piece that is not the last piece and that contains at least two edges. The total length of all edges except the last is at most A_p, and the length of the last edge is less than $2A_p$. So, its total length is less than $3A_p \leq \omega_{\max}$.
- If we placed less than k valves, the analysis of the last piece is the same as in the previous two cases.

It is straightforward that the algorithm uses $O(n)$ time. □

3.2 Cycles

If G is a cycle, then we can obtain exact and approximate solutions for the VALVE LOCATION by using variants to the algorithms for paths.

If we have k valves, and know the location of one of the valves on a cycle, then the problem reduces to solving the VALVE LOCATION problem on a path, where we can place $k-1$ valves. Trying each vertex as location for the first valve gives algorithms for cycles that use n times the time for an algorithm on paths.

A small trick helps to limit the number of choices for the first valve, and thus to reduce the running time on cycles. We first look at a fast 2-approximation algorithm.

Proposition 6 *The* VALVE LOCATION *problem on a cycle admits a 2-approximation algorithm that uses $O(n)$ time.*

Proof. Write $A_c = \omega_\Sigma/k$. Use the following heuristic. Place the first valve at some arbitrary vertex v. Then, place the other $k-1$ valves similar to the 2-approximation algorithm for paths: starting from the last location where we have put a valve (in the first round v), walk along the cycle till we have a piece whose total length is at least A_c, and then place a valve. After we have placed k valves, the remaining piece has total length at most $\omega_\Sigma - (k-1)A_c \leq A_c$. Each other piece has length at most $A_c + \omega_{\max}$. As A_c and $\omega_{\max}$ both are lower bounds for the optimal value, we have a 2-approximation. (With k valves, the cycle is divided in k pieces, and thus there always will be a piece of total length at least ω_Σ/k.) □

Proposition 7 *Consider the* VALVE LOCATION *problem on a cycle. Let k be the number of valves. If $\omega_{\max} \geq 3\omega_\Sigma/k$, then the optimal spill equals $\omega_{\max}$. An optimal valve location can be found in this case in $O(n)$ time.*

Proof. The proof is similar to the proof of Proposition 5. Again, write $A_c = \omega_\Sigma/k$.

Put two valves at both endpoints of the edge with length $\omega_{\max}$. Then, put the other valves similar as in the proof of Proposition 5, ending a piece when it has total length at least A_c, or when the next edge has length at least $2A_c$.

If we placed k valves, then the last piece has length at most A_c: we have one piece of length $\omega_{\max} \geq 3A_c$, and each of the other $k-2$ pieces either has total length at least A_c, or has length at most A_c, but is followed by a single edge piece of length at least $2A_c$. So, the other pieces have total length at least $3A_c + (k-2)A_c$.

An analysis, similar to the proof of Proposition 5, shows that each other piece has length at most $\omega_{\max}$. □

Theorem 8 *The* VALVE LOCATION *problem on a cycle can be solved by solving $O(n/k)$* VALVE LOCATION *problems on paths of length at most n.*

Proof. Use $A_c = \omega_\Sigma/k$, as before. If $\omega_{\max} \geq 3A_c$, then by Proposition 7 there is an optimal valve placement that places two valves on the endpoints of the longest edge. Place

these valves, and solve the problem of placing the last $k-2$ valves on the remaining path optimally by one call to the VALVE LOCATION problem on a path.

If $\omega_{\max} < 3A_c$, then there is a solution of value $\omega_{\max} + A_c < 4A_c$. Now, partition the cycle into $\lfloor k/4 \rfloor$ segments: each segment is a path on the cycle; we do this such that the number of edges on different segments differ by at least one. So, each segment has $O(n/k)$ edges. At least one of these segments must have total length at least $4\omega_\Sigma/k = 4A_c$, which is larger than the optimal value. So, for the segment with the largest total length, we know that at least one of its interior vertices has a valve in an optimal solution.

Thus, our algorithm runs as follows: find the heaviest segment, and try all $O(n/k)$ placements of a valve on an interior vertex of the segment. For each such placement, the remaining problem is equivalent to placing $k-1$ valves on a path. □

Corollary 9 *The* VALVE LOCATION *problem on a cycle can be solved in* $O(n \cdot \min\{\log \omega_\Sigma, n/k\})$ *time.*

3.3 Trees

In this section, we show that the problem on trees can be solved in polynomial time. More specifically, we show:

Theorem 10 *The* VALVE LOCATION *problem on a tree can be solved in* $O(nk^2 \log(n\omega_{\max}))$ *time.*

The global structure of the algorithm is a binary search on the optimal value in the range of integers between 0 and $n\omega_{\max}$. Thus, we directly obtain Theorem 10 as a corollary of the next result.

Proposition 11 *Given a tree, and an integer* L*, we can decide in* $O(nk^2)$ *time if we can place* k *valves with maximum piece size at most* L*.*

Proof. We choose an arbitrary vertex v_r as root of the tree. For rooted subtrees T', and integers i, $0 \le i \le k$, we define

- $A_{T',L}(i)$ = the minimum over all possible ways to put at most i valves in T' such that no piece in T' has a total length of more than L, of the total length of the piece that contains the root node of T'.
- $A_{T',L}(i) = 0$, if there is a way to put at most i valves in T' such that no piece in T' has a total length of more than L, such that there is a valve in the root node of T'.
- $A_{T',L}(i) = \infty$, if there is no possible way to put at most i valves in T' such that no piece in T' has a total length of more than L.
- $P_{T',L}(i) =$ **true** if and only if $A_{T',L}(i) = 0$, i.e. if we can put at most i valves in T' such that no piece in T' has a total length of more than L, such that there is a valve in the root node of T'.

We will compute tables $A_{T',L}$ and $P_{T',L}$ for several subtrees of T:

- For each vertex v in T except v_r, we compute a table for the subtree, consisting of the parent of v in T, v, and all the descendants of v. The root of this subtree is the parent of v. Call this subtree T_v^+.
- For each vertex v in T: if v has i children $w_1, w_2, \ldots, w_i$, then for each j, $0 \leq j \leq i$, we compute a table for the subtree, consisting of v, $w_1, \ldots, w_j$, and all descendants of $w_1, w_2, \ldots, w_j$. Vertex v is the root of this subtree. Call this subtree $T_{v,j}$. For the case $j = i$, write $T_v = T_{v,i}$; this is the tree consisting of v and all its descendants.

The following two lemmas give recursive formulations that show how to compute these tables.

Lemma 12 *Let T be obtained by taking the union of trees T' and T'' such that the root r of T' and T'' is the only vertex that belongs to both trees. Let $0 \leq i \leq k$.*

1. *$P_{T,L}(i)$, if and only if there are i', i'' with $i' + i'' = i - 1$, $0 \leq i' \leq k$, $0 \leq i'' \leq k$, such that $P_{T',L}(i')$ and $P_{T'',L}(i'')$.*
2. *If $P_{T,L}(i)$, then $A_{T,L}(i) = 0$.*
3. *If not $P_{T,L}(i)$, then*

$$A_{T,L}(i) = \min_{i',i'',i'+i''=i,0\leq i,i'} A_{T',L}(i') + A_{T'',L}(i'')$$

if this term is at most L, otherwise $A_{T,L}(i) = \infty$.

Proof. 1. If $P_{T,L}(i)$, then consider a placement of the valves in T, with a valve in r such that each piece has total length at most L. Suppose i' of these valves are in T', and i'' of these valves are in T''. As exactly one valve belongs to both T' and T'' (namely, the valve in r), $i = i' + i'' - 1$. The placement of the valves in T' shows that $P_{T',L}(i')$, and similarly, $P_{T'',L}(i'')$.

Suppose there are i' and i'' with $i' + i'' = i - 1$, $0 \leq i' \leq k$, $0 \leq i'' \leq k$, such that $P_{T',L}(i')$ and $P_{T'',L}(i'')$. Combine the placement of i' valves in T' (with pieces of length at most L) with a placement of i'' valves in T'' (with pieces of length at most L), each with a valve in the root vertex. As these placements share the root vertex, we have exactly $i' + i'' - 1 = i$ valves, and each piece has a total length at most L. Hence $P_{T,L}(i)$.

2. It is trivial that if $P_{T,L}(i)$, then $A_{T,L}(i) = 0$.

3. Suppose that $P_{T,L}(i)$ is **false**. Suppose we have a placement of i valves in T such that each piece has total length at most L, and the length of the piece containing r is minimized. Now, we cannot have placed a valve in r (otherwise $P_{T,L}(i)$ holds). Suppose i' of the valves are in vertices of T', and i'' in vertices of T''. As there is no valve in r, we have that $i' + i'' = i$. Consider the piece that contains r. Suppose α is the length of all edges

of this piece in T', and β is the length of all edges of this piece in T''. Now, $\alpha \geq A_{T',L}(i')$, $\beta \geq A_{T'',L}(i'')$, and $A_{T,L}(i) = \alpha + \beta$. So,

$$\min_{i',i'',i'+i''=i,0\leq i',i''} A_{T',L}(i') + A_{T'',L}(i'') \leq L$$

and

$$A_{T,L}(i) \geq \min_{i',i'',i'+i''=i,0\leq i',i''} A_{T',L}(i') + A_{T'',L}(i'').$$

Consider some i', i'' with $i' + i'' = i$. Suppose there is a placement of i' valves in T' with each piece of total length at most L, and the piece containing r of total length $A_{T',L}(i')$. Similarly, suppose there is a placement of i'' valves in T'' with each piece of total length at most L, and the piece containing r of total length $A_{T'',L}(i'')$. Neither of these placements can put a valve in r; otherwise, the union of the placements gives $P_{T,L}(i)$ since we place at most i valves, have a valve in r, and each piece has total length at most L. Hence, in the union of the placements, the resulting piece that contains r contains edges in T' and edges in T''; its total length equals $A_{T',L}(i') + A_{T'',L}(i'')$. Thus, if $A_{T',L}(i') + A_{T'',L}(i'') \leq L$, we have

$$A_{T,L}(i) \leq A_{T',L}(i') + A_{T'',L}(i'').$$

Hence,

$$A_{T,L}(i) \leq \min_{i',i'',i'+i''=i,0\leq i',i''} A_{T',L}(i') + A_{T'',L}(i''),$$

if this term is at most L. The result now follows. □

Lemma 13 *Let T be a tree with root r, and let T^+ be obtained by adding an edge $\{r, r'\}$ to a new vertex r' with length ℓ. Let r' be the root of T^+. Let $0 \leq i \leq k$, and L be an integer.*

1. *$P_{T^+,L}(i)$ holds if and only if $i > 0$ and $A_{T,L}(i-1) + \ell \leq L$.*
2. *If $P_{T^+,L}(i)$, then $A_{T^+,L}(i) = 0$.*
3. *If not $P_{T^+,L}(i)$, then $A_{T^+,L}(i) = A_{T^+,L}(i) + \ell$, if this term is at most L, and $A_{T^+,L}(i) = \infty$ otherwise.*

Proof. 1. Suppose $P_{T^+,L}(i)$ is **true**. As we place a valve in r', we have $i > 0$. Consider the placement of the (at most) $i-1$ remaining valves in T. If it has a valve in r, then $A_{T,L}(i-1) = 0$ and $\ell \leq L$. Otherwise, the piece in T' containing r has total length at most L, so if we take the edge $\{r, r'\}$ out of this piece, we obtain a placement of $i-1$ valves in T with the piece containing r of total length at most $L - \ell$. Hence $A_{T,L}(i-1) \leq L - \ell$.

Suppose $i > 0$ and $A_{T,L}(i-1) + \ell \leq L$. Consider a placement of $i-1$ valves in T such that each piece has total length at most L, and the piece containing r has length $A_{T,L}(i-1) \leq L - \ell$. Take these valves in T', and add a valve in r. Observe now that each piece has total length at most L.

2. Trivial.

3. Suppose $P_{T^+,L}(i)$ is **false**. If there is also no placement of i valves in T such that each piece has total length at most L, then $A_{T^+,L}(i) = A_{T^+,L}(i) = \infty$. Otherwise, take a placement of i valves such that each piece has total length at most L, and the piece containing r has length $A_{T,L}(i)$, or $A_{T,L}(i) = 0$ and there is a valve in r. Taking the same valves in T^+ gives a placement of valves with each piece except the piece containing r' having total length at most L, and the piece containing r' having total length $A_{T,L}(i) + \ell$. So $A_{T^+,L}(i) \leq A_{T^+,L}(i) + \ell$, in case $A_{T^+,L}(i) + \ell \leq L$.

Suppose we have a placement of i valves in T^+ such that each piece has total length at most L, and the length of the piece containing r' is minimized. By assumption, there is no valve in r'. If there is a valve in r, then $A_{T,L}(i) = 0$ and $A_{T^+,L}(i) = \ell$. Otherwise, we get a minimum total length piece containing r', if the restriction of this piece to T has minimum total length, i.e., has length $A + T, L(i)$. Hence, $A_{T^+,L}(i) = A_{T^+,L}(i) + \ell$. □

Using Lemmas 12 and 13, we can compute all desired tables. Recall that L is fixed during the computation. Now, for all vertices in the tree, in postorder, we compute all values $A_{T_v,L}(i)$, and $P_{T_v,L}(i)$, for all i, $0 \leq i \leq k$. This is done in the following way. If v is a leaf of T, then computing these values is trivial. Otherwise, suppose v has s children, say $w_1, w_2, \ldots, w_s$. For all j, $1 \leq j \leq s$, we compute all values $A_{T_{v,j},L}(i)$, and $P_{T_{v,j},L}(i)$ for all i, $0 \leq i \leq k$. In case $j = 1$, we note that $T_{v,1}$ is the same subtree as $T^+_{w_1}$. Thus, using Lemma 13, we can compute the values $A_{T_{v,1},L}(i)$, and $P_{T_{v,1},L}(i)$ from the already earlier computed tables $A_{T_{w_1},L}$ and $P_{T_{w_1},L}$. For $2 \leq j \leq s$, we note that $T_{v,j}$ is the union of $T_{v,j-1}$ and $T^+_{w_j}$. Thus, we first compute the tables $A_{T^+_{w_j}}, L$ and $P_{T^+_{w_j}}, L$ given the tables $A_{T_{w_j},L}$ and $P_{T_{w_j},L}$ using Lemma 13. Then, we compute the tables $A_{T_{v,j},L}$ and $P_{T_{v,j},L}$ from the tables $A_{T^+_{w_j},L}$, $P_{T^+_{w_j},L}$, $A_{T_{v,j-1},L}$ and $P_{T_{v,j-1},L}$, using Lemma 12. Finally, note that $T_{v,s} = T_v$. When we have the tables $A_{T_{v_r},L}$ and $P_{T_{v_r},L}$, we can easily decide whether we can place k valves in T with maximum piece size at most L, using the following simple observation.

Proposition 14 *Let v_r be the root of T. There is a solution to the* VALVE LOCATION *problem with k valves and cost at most L if and only if $A_{T_{v_r},L}(k) < \infty$.*

If T has n vertices, then we compute $O(n)$ tables: $O(1)$ per edge in T. Each table can be computed in $O(k^2)$ time. This can be easily observed from Lemmas 12 and 13. Simply, iterate over all possible values of k and k', and compute the necessary value of k''. Each step involves $O(1)$ computations. Actually, the step that uses Lemma 13 needs only $O(k)$ time. □

For trees, a result similar to Propositions 5 and 7 holds. However, in this case, this does not lead to an approximation algorithm with constant performance guarantee.

Proposition 15 *Consider the* VALVE LOCATION *problem on a tree. Let k be the number of valves. If $\omega_{\max} \geq 3\omega_\Sigma/k$, then the optimal spill equals $\omega_{\max}$.*

Proof. Write $A_T = \omega_\Sigma/k$. We give an algorithm that realizes a spill of $\omega_{\max}$. During the algorithm, we place some valves and allocate the not yet placed valves to pieces, i.e. maximal subtrees without an already placed valve. Initially, we have no valves placed and allocate all valves to the entire tree. As an invariant, a piece of total length ℓ has at least $\lfloor \ell/A_T \rfloor$ valves allocated to it, or has total length at most $\omega_{\max}$. This clearly holds initially.

If a piece with at least two edges contains an edge e with length at least $2A_T$, then we place a valve at each endpoint of this edge. Suppose the piece has length ℓ. Thus, $\lfloor \ell/A_T \rfloor$ valves were allocated to this piece, of which we used two. Hence, we can allocate remaining $\lfloor \ell/A_T \rfloor - 2$ valves to the created subpieces, except the piece with the single edge e. To each subpiece i with total length ℓ'_i, we allocate $\lfloor \ell'_i/A_T \rfloor$ valves. As $\sum_i \ell'_i \leq \lfloor (\ell - 2A_T)/A_T \rfloor = \lfloor \ell/A_T - 2 \rfloor = \lfloor \ell/A_T \rfloor - 2$, we have sufficient number of valves for the allocation.

Now, suppose each piece with at least two edges only has edges with length less than $2A_T$. If a piece has total length at most $\omega_{\max}$, then we place the valves allocated to the piece arbitrarily. Consider now a piece p whose total length ℓ is more than $\omega_{\max}$. Take an arbitrary vertex in the piece v_r, and look at the piece as a rooted tree, with root v_r. For each vertex x in the piece, let $w(x)$ be the total length of all edges 'below' x, i.e. in the subtree rooted at x. Note that $w(v_r) = \ell$.

First, suppose there is a vertex z, with $A_T \leq w(z) \leq \omega_{\max}$. Now, place a valve at z. This splits the piece in one or more pieces, containing edges with a descendant of z, and one other piece, containing amongst others v_r. We allocate no valves to the first collection of pieces, and all remaining $\lfloor \ell/A_T \rfloor - 1$ valves to the latter subpiece, whose total length is at most $\ell - A_T$. So, the invariant is kept intact.

Now, suppose for all vertices z, either $w(z) < A_T$ or $\omega_{\max} < w(z)$. Take the following walk in the piece. Start at $w(v_r)$, and always move to the child y with maximal value $w(y)$. Stop when we arrive in a vertex y with $w(y) < A_T$. Let z be the parent of y. We must have $w(z) > \omega_{\max}$. Now, we place a valve at z. Again, we allocate all remaining $\lfloor \ell/A_T \rfloor - 1$ valves to the subpiece that contains v_r. Possibly, $v_r = z$, and then we do not need to allocate additional valves. This piece has total length $\ell - w(z) \leq \ell - w(z) < \ell - A_T$, so we allocated sufficiently many valves to it. Consider a resulting subpiece that contains child y' of z. As y was the child with maximal value $w(y)$, we have that $w(y') \leq w(y) < A_T$. Hence, the total length of the edges in this subpiece is at most $w(y') + \omega_{\{y',z\}} \leq A_T + 2A_T \leq \omega_{\max}$.

We repeat this process until all valves are placed. Then, no piece has a total length more than $\omega_{\max}$. □

4 Algorithms for Graphs of Bounded Treewidth

In practice, the pipe systems are more complicated than trees. This makes the problem more difficult from algorithmic perspective. Fortunately, most of the real-life pipe networks are outerplanar or, taking this more generally, the corresponding graphs have bounded treewidth; see e.g. [5, 6, 10]. For this type of networks we have the following results.

Theorem 16 *The* VALVE LOCATION *problem on graphs of treewidth q admits a dynamic programming algorithm running in time $(n\omega_{\max})^{O(q)}$.*

This dynamic programming algorithm follows the lines of several algorithms for other problems on graphs of bounded treewidth. For easier description, we use a *nice* tree decomposition of width at most q; for definition see below.

As a first step, we must find a tree decomposition of width at most q. This can be done in $O(n)$ time for fixed q; see [4]. At this point, we would like to make a remark concerning practical implementations. The algorithm in [4] has such a large hidden constant, that it is not of use in a practical setting. Fortunately, there are several heuristics that often give good bounds. Also, there are fast algorithms that construct tree decompositions of optimal width for graphs of treewidth at most three (including outerplanar graphs), see e.g. [6] for a discussion.

Given a tree decomposition, in $O(n)$ time one can transform it to a *nice* tree decomposition [11] with the same width. We now give the definition of a nice tree decomposition.

A *nice tree decomposition* of a graph $G = (V, E)$ is a rooted binary tree $T = (I, F)$, where each node $i \in I$ is a subset $X_i \subseteq V$, called *bag*, such that

1. $\bigcup_{i \in I} X_i = V$.
2. For all $\{v, w\} \in E$, there exists an $i \in I$, with $v, w \in X_i$.
3. For all $v \in V$, the set $\{i \in I \mid v \in X_i\}$ forms a subtree of T.
4. If $i \in I$ has two children j_1, j_2, then $X_i = X_{j_1} = X_{j_2}$ (Join Node).
5. If $i \in I$ has one child j, then either there is a $v \in X_i$ with $X_j \cup \{v\} = X_i$ (Introduce Node) or there is a $v \in X_j$ with $X_i \cup \{v\} = X_j$ (Forget Node).
6. If $i \in I$ is a leaf in T, then $|X_i| = 1$ (Leaf Node).

The *width* of a nice tree decomposition is $\max_{i \in I} |X_i| - 1$.

In our dynamic programming algorithm, we compute in postorder for each node of T a table. Associate to node $i \in I$ the subgraph $G_i = G[V_i]$, induced by the set of vertices in X_i or a bag X_j with j a descendant of i: $V_i = \bigcup X_j$, with the union taken over all j in the subtree of T rooted at j.

A placement of valves on the vertices of G_i has a *characteristic*, which is a 5-tuple $(j, Z, L, f, \sim)$, consisting of

- The number j of used valves in G_i.
- The subset $Z \subset X_i$ of the vertices in X_i that contain a valve.
- The maximum length of a piece in G_i.
- A function $f : X_i \to \mathbf{N}$, giving for each vertex $v \in V_i$ the total length of the piece that contains v; if there is a valve on v, then $f(v) = 0$.
- An equivalence relation $\sim$ on X_i, with for all $v, w \in X_i$, $v \sim w$, if and only if there is a path from v to w in G_i that does not contain a vertex with a valve.

In the table of i, we store all possible characteristics of all placements of valves in G_i. Note that in this way, tables have a size that is bounded by $(n\omega_{\max})^{O(q)}$.

A somewhat tedious case analysis, typical for dynamic programming algorithms on graphs of bounded treewidth, shows that we can compute for each of the four types of nodes the table of all characteristics for a node, given such a table for each of the children of the node, in time polynomial in the table size.

Then, computing these tables for all nodes in postorder gives an algorithm computing the table for the root node, and as G_r for the root node r equals G, we obtain the optimal valve location from this table.

We remark that the described dynamic programming is only a pseudo-polynomial time algorithm for the weighted version of the VALVE LOCATION problem on graphs of bounded treewidth. Using standard scaling arguments, we derive the following corollary.

Corollary 17 *The* VALVE LOCATION *problem on graphs of bounded treewidth admits a fully polynomial approximation scheme.*

5 Complexity Results

In this section, we show that two restricted versions of the VALVE LOCATION problem are NP-hard. For general networks it is strongly NP-hard as even the unweighted version of the problem is NP-hard, while for series-parallel graphs (a special case of graphs of treewidth at most two) the problem is weakly NP-hard. Note that this complements the result that the problem is solvable in pseudo-polynomial time on graphs of bounded treewidth.

Theorem 18 *The* VALVE LOCATION *problem is NP-hard even if* $\omega_e = 1$ *for all* $e \in E$.

Proof. The proof of this theorem uses a reduction from the strongly NP-hard problem 3-PARTITION, see e.g. [8], where, given a set A of $3m$ elements, a bound $B \in Z^+$, and a size $s(a) \in Z^+$ for each $a \in A$ such that $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$, the questions is: Can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$?

Given an instance of 3-PARTITION, we construct an instance of the unweighted VALVE LOCATION problem as follows. For each element $a \in A$ we create a subgraph H_a in G, for illustration see Figure 1:

- First, let H_a contain a clique $K_\xi(a)$ on ξ vertices where ξ is large enough, e.g. $\xi = 9m^2 s_{\max}$.
- Add to H_a an apex-vertex v_a adjacent to any $s(a) \cdot \phi \leq \xi$ vertices in the clique $K_\xi(a)$. Here, ϕ is also large enough, e.g. $\phi = 9m^2$.
- In addition, for the entire graph G introduce $3m(3m-1)/2$ transit-vertices $v_{\{a,b\}}$ for each pair of distinct elements a and b from A. For each such a pair, make vertex

$v_{\{a,b\}}$ adjacent to all vertices in cliques $K_\xi(a)$ and $K_\xi(b)$. Let vertex $v_{\{a,b\}}$ belong to both subgraphs H_a and H_b. Let all edges connecting $v_{\{a,b\}}$ to clique $K_\xi(a)$ belong to H_a and all edges connecting $v_{\{a,b\}}$ to clique $K_\xi(b)$ belong to H_b.

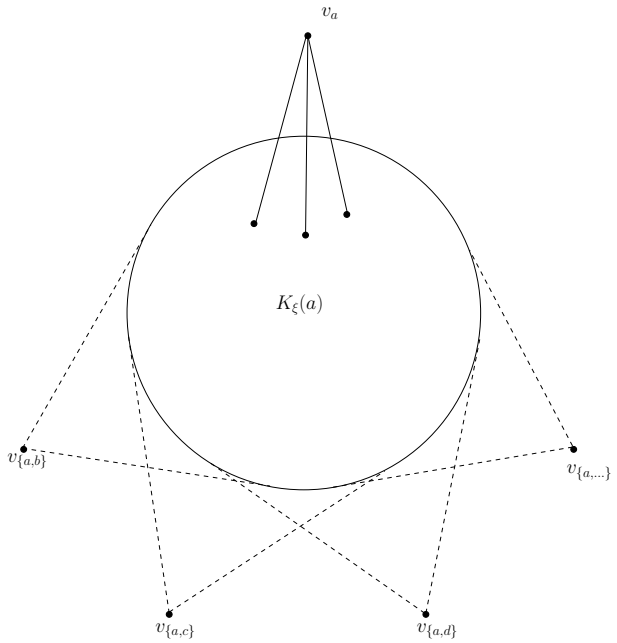

Figure 1: Subgraph $H_a,\ a \in A$

Notice that, by construction, the subgraphs are intersecting only in the set of transit-vertices. Now, when graph $G = (V, E)$ is constructed, we complete specification of the instance of the VALVE LOCATION problem defining the number of valves $k = \frac{3m(3m-1)}{2} - 3m = 9m(m-1)/2$ and the edge weights $\omega_e = 1$ for all $e \in E$. Clearly, under unary encoding of sizes $s(a), a \in A$, the presented reduction is polynomial.

Now, to prove the theorem, we show that, given an instance of 3-PARTITION, there is a solution to that instance if and only if there exists a solution to the corresponding instance of the VALVE LOCATION problem with maximum spill at most $\frac{3\xi(\xi-1)}{2} + 3(3m-1)\xi + B\phi$.

For 'only if' direction, assume that there is a partition of A into m disjoint sets $A_1, A_2, \dots, A_m$ such that, for $1 \le i \le m$, $\sum_{a \in A_i} s(a) = B$. Put the valves in all vertices $v_{\{a,b\}}$ of G such that a and b are not from the same set A_i. Thus we partition the edge set of G into m pieces corresponding to three elementary sets $A_1, A_2, \dots, A_m$. Consider a piece corresponding to a set $A_i = \{a, b, c\}$, see Figure 2. This piece consists of a union of subgraphs $H_a,\ H_b$ and H_c. More precisely, it consists of (i) three cliques on ξ vertices $K_\xi(a),\ K_\xi(b)$ and $K_\xi(c)$ that is $\frac{3\xi(\xi-1)}{2}$ edges; (ii) three sets of edges from the cliques to apex-vertices $v_a,\ v_b$, and v_c that is $s(a)\phi + s(b)\phi + s(c)\phi = B\phi$ edges; and three sets of edges connecting the cliques to the valves and to vertices $v_{\{a,b\}},\ v_{\{a,c\}}$ and $v_{\{a,b\}}$ that is $3(3m-1)\xi$ edges. The claimed maximum spill follows straightforwardly.

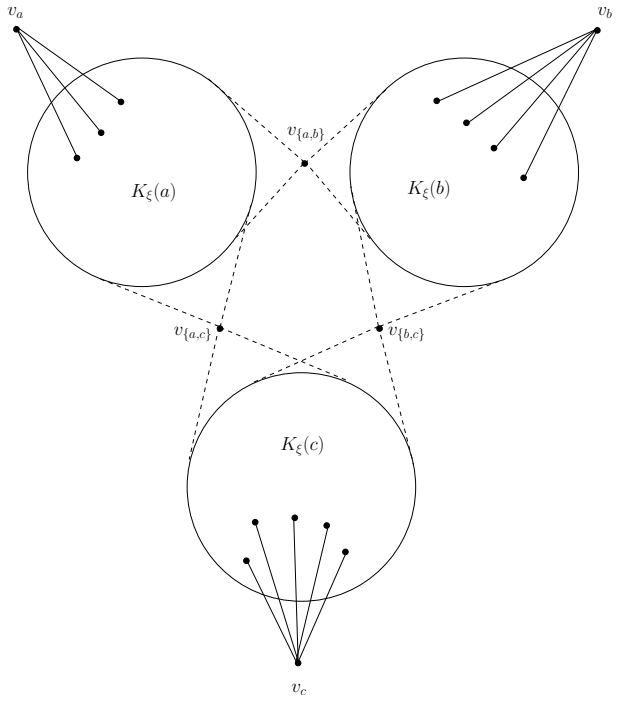

Figure 2: A piece of the partition of G

For 'if' direction, assume that to the constructed instance of the VALVE LOCATION problem there is a solution with maximum spill $\frac{3\xi(\xi-1)}{2}+3(3m-1)\xi+B\phi$. First, we observe that in any optimal solution to the constructed instance the valves will be installed only in transit-vertices of G and only in such a way that G is partitioned into m pieces where each piece consists of three subgraphs corresponding to some three elements of A. For otherwise, since $\xi \geq \phi \gg k$, there is a piece containing roughly four subgraphs that contradicts to the assumed upper bound on the maximum spill. Then, similarly to the analysis above, we arrive to the conclusion that each piece has the number of edges connecting the cliques to the apex-vertices equal to $s(a)\phi + s(b)\phi + s(c)\phi = B\phi$. This yields that three elementary subsets $A_1, A_2, \ldots, A_m$ of the corresponding partition of A have the same size B that completes the proof. □

For the second complexity result, let us remind a definition of a series-parallel graph. A *series-parallel graph* is a graph $G = (V, E)$ with two special vertices, called its terminals, often denoted s and t, that can be formed with the following operations:

- A graph consisting of a single edge $\{s, t\}$ between its terminals is a series-parallel graph.
- If G and H are terminal graphs, with terminals s_G, t_G, and s_H and t_H, then the *series composition* of G and H is a series-parallel graph. In the series composition,

we take the disjoint union, then identify t_G and s_H, and take s_G and t_H as terminals of the resulting graph.

- If G and H are terminal graphs, with terminals s_G, t_G, and s_H and t_H, then the *parallel composition* of G and H is a series-parallel graph. In the parallel composition, we take the disjoint union, then identify s_G and s_H and identify t_G and t_H. The two vertices obtained by identification are the terminals of the resulting graph.

Theorem 19 *The* VALVE LOCATION *problem is weakly NP-hard for series-parallel graphs.*

Proof. We show that the VALVE LOCATION problem is weakly NP-hard for the following graphs: we have two vertices s and t, and a number of internally disjoint paths from s to t of length exactly five.

We use a reduction from PARTITION; see e.g. [8]. Suppose we are given positive integers $a_1, a_2, \ldots, a_n$. The PARTITION problem asks if these integers can be partitioned into two sets with equal sum, i.e. we look for two sets, each of sum $B = \sum_{i=1}^{n} a_i/2$. We may assume B is integer, as if $\sum_{i=1}^{n} a_i$ is odd, the PARTITION problem trivially has no solution.

As the corresponding instance for the VALVE LOCATION problem, we take n disjoint paths from s to t. Each of these paths has length five, i.e., four intermediate vertices, which we call $v_{i,1}, v_{i,2}, \ldots, v_{i,4}$. The successive lengths of the edges on the ith path are 1, a_i, $B - a_i + n$, a_i, 1. Call the resulting graph G, see Figure 3.

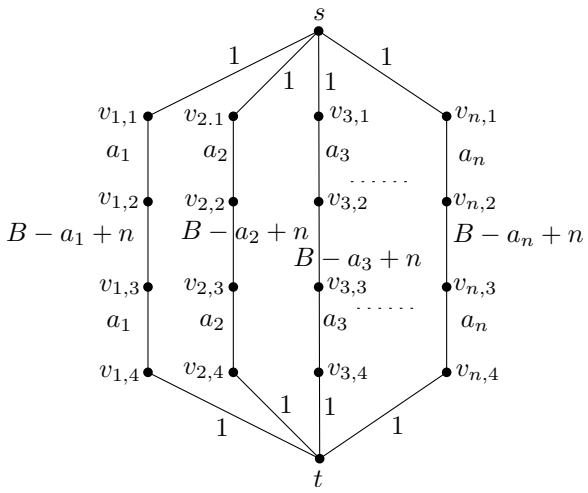

Figure 3: The series-parallel graph constructed in Theorem 19

Proposition 20 *Set $A = \{a_1, a_2, \ldots, a_n\}$ can be partitioned into two sets, both of sum B, if and only if we can place at most $2n$ valves in G such that each part has total length at most $B + n$.*

Proof. Suppose set A can be partitioned into sets S_1 and S_2, both of sum B. Now, for each i, $1 \leq i \leq n$, $a_i \in S_1$, put valves on $v_{i,2}$ and $v_{i,4}$. Otherwise, i.e., $a_i \in S_2$, put valves on $v_{i,1}$ and $v_{i,3}$. Each piece has total length at most $B + n$. We have some pieces, consisting of two edges on a path; these all have total length exactly $B + n$. The piece containing s has n edges of length 1, and for all $a_i \in S_1$, an edge of length a_i, and hence has total length $n + \sum_{a_i \in S_1} a_i = B + n$. Similarly, the piece containing t has total length $n + \sum_{a_i \in S_2} a_i = B + n$.

Suppose we can place $2n$ valves such that each piece has total length at most $B + n$. First, note that for each i, at least two valves must be placed on vertices $v_{i,1}, \ldots, v_{i,4}$: if we use at most one, then at least three edges on the ith path belong to the same piece, and each part of three successive edges on this path has a total length that is more than $B+n$. As we have $2n$ valves in total, we must use exactly two valves on each set of vertices $v_{i,1}, \ldots, v_{i,4}$, and cannot place a valve on s or on t. In addition, we cannot place a valve on both a vertex $v_{i,1}$ and a vertex $v_{i,4}$ for some i, as this would give a piece with length more than $B + n$.

Now we define sets S_1 and S_2 as follows. For each i, $1 \leq i \leq n$, place a_i in S_1 if there is no valve on $v_{i,1}$, otherwise place a_i in S_2. The piece containing s contains n edges of length 1, and for each $a_i \in S_1$ an edge of length a_i. Hence, $n+\sum_{a_i \in S_1} a_i \leq B+n$, so $\sum_{a_i \in S_1} a_i \leq B$. For each $a_i \in S_2$, we have a valve on $v_{i,1}$, and hence no valve on $v_{i,4}$, and hence the piece containing t contains the edge $\{v_{i,3}, v_{i,4}\}$ with length a_i. So, $n + \sum_{a_i \in S_2} a_i \leq B + n$, by considering the total length of the piece containing t, and hence $\sum_{a_i \in S_2} a_i \leq B$.

As we now have a partition of A into two disjoint sets S_1, S_2 with $\sum_{a_i \in S_1} a_i \leq B$, and $\sum_{a_i \in S_2} a_i \leq B$, we have $\sum_{a_i \in S_1} a_i = \sum_{a_i \in S_2} a_i = B = \sum_{i=1}^{n} a_i/2$. □

The NP-hardness of the VALVE LOCATION problem on series-parallel graphs now follows, by noting that G is series-parallel: a path can be constructed by a sequence of series compositions, and by parallel compositions, we can identify the endpoints of the paths. □

As series-parallel graphs have treewidth two, the results of the previous section show that (unless P=NP), the problem on series-parallel graph cannot be strongly NP-hard.

6 Conclusions, extensions and open questions

In this paper we presented fast algorithms for several practically relevant classes of instances of the VALVE LOCATION problem. Moreover, applying literally the same techniques to the well known vertex integrity problem, we can tackle this later problem as well. For the vertex integrity problem we have to redefine the spill as the maximum number of vertices in a component instead of the maximum number of edges in a component. Given a number of valves $1 \leq k \leq n$, to find the redefined spill we can straightforwardly adjust the algorithms above. Finally, to find the vertex integrity it is sufficient to find a minimizer of the the sum of k and the corresponding optimal spill over all possible values for k : $1 \leq k \leq n$.

This means that at additional cost of factor n in the running time we can solve the vertex integrity problem.

A number of interesting problems, however, remains open:

- Can we apply somewhat similar techniques to more complicated cost and network vulnerability measures, see e.g. [9, 14]?
- What is an approximability status of the VALVE LOCATION problem on general planar graphs?
- Consider a continuous, purely geometric, version of the VALVE LOCATION problem. Given is a set of rectilinear curves in a plane, possibly intersecting each other. We allow to place a valve in any point of any curve. The cost (the spill) of a piece is defined now as the total Euclidean length of a piece. The same question arises: how to place k valves on this continuous network such that the maximum piece length is minimized? Complexity and approximability of this very natural problem is also open.

Acknowledgments

We thank Alexandr Kostochka for pointing on several very useful references on graph integrity.

References

[1] N. Alon, P. Seymour and R. Thomas. A separator theorem for graphs with an excluded minor and its applications. In Proc. of the 22nd Symposium on Theory of Computing, STOC'80, pages 293-299. ACM Press, 1980.

[2] C.A. Barefoot, R. Entringer and H.C. Swart. Vulnerability in graphs: a comparative survey. J. Comb. Math. Comb. Comput., 1:12-22, 1987.

[3] C.A. Barefoot, R. Entringer and H.C. Swart. Integrity of trees and the diameter of a graphs, Congressus Numerantium, 58:103-114, 1987.

[4] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput., 25:1305–1317, 1996.

[5] H.L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. Theor. Comp. Sc., 209:1–45, 1998.

[6] H.L. Bodlaender. Treewidth: Characterizations, applications, and computations. In F. V. Fomin, editor, Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science, WG'06, pages 1-14. Springer Verlag, Lecture Notes in Computer Science, vol. 4271, 2006.

[7] U. Feige and M. Mahdian. Finding small balanced separators. In Proceedings of the 37th Annual Symposium on Theory of Computing, STOC'06, pages 375-384. ACM Press, 2006.

[8] M.R. Garey and D.S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman, San Francisco, 1979.

[9] N.V. Grigorieva and A. Grigoriev. Optimal valve location in long oil pipelines. Research Memorandum RM/07/007, Maastricht Research School of Economics of Technology and Organizations (METEOR), Maastricht University, Maastricht, The Netherlands, 2007. World Wide Web: http://ideas.repec.org/p/dgr/umamet/2007007.html

[10] I.V. Hicks, A.M.C.A. Koster, and E. Kolotoglu. Branch and tree decomposition techniques for discrete optimization. In J.C. Smith, editor, TutORials 2005, INFORMS Tutorials in Operations Research Series, chapter 1, pages 1-29. INFORMS Annual Meeting, 2005.

[11] T. Kloks. *Treewidth. Computations and Approximations*. Lecture Notes in Computer Science, Vol. 842. Springer-Verlag, Berlin, 1994.

[12] D. Kratsch, T. Kloks and H. Müller. Measuring the vulnerability for classes of intersection graphs. Discrete Applied Mathematics, 77(3):259–270, 1997.

[13] D. Marx. Parameterized Graph Separation Problems. In R.G. Downey, M.R. Fellows and F.K.H.A. Dehne, editors, Proceedings of the First International Workshop on Parameterized and Exact Computation, IWPEC'04, pages 71–82. Springer Verlag, Lecture Notes in Computer Science, vol. 3162, 2004.

[14] S. Ozger and L.W. Mays. Optimal location of isolation valves: A reliability approach. In Water Supply Systems Security, Digital Engineering Library, McGraw-Hill, 2004. World Wide Web: http://dx.doi.org/10.1036/0071455663.CH13